University System by Programmers' Paradise

23 December, 2024

Satybalbyeva Gaukhar
Saparbay Nursaya
Abuov Asanali

```java
public Manager(String lastName, String firstName, ManagerType type) {
    super(lastName, firstName);
    this.type = type;
}

public ManagerType getType() {
    return type;
}

public void setType(ManagerType type) {
    this.type = type;
}

@Override
public int hashCode() {
    final int prime = 31;
    int result = super.hashCode();
    result = prime * result + Objects.hash(type);
    return result;
}

@Override
public boolean equals(Object obj) {
    if (!super.equals(obj)) return false;
    Manager other = (Manager) obj;
```

Manager

```java
public Employee() {}

public Employee(String firstName, String lastName, double salary) {
    super(firstName, lastName);
    this.salary = salary;
    this.hireDate = new Date();
}

public Employee(Integer id, String firstName, String lastName, String email, String password, double salary) {
    super(id, firstName, lastName, email, password);
    this.salary = salary;
    this.hireDate = new Date();
}

// Геттеры и Сеттеры
public double getSalary() {
    return salary;
}

public void setSalary(double salary) {
    this.salary = salary;
}

public Date getHireDate() {
    return hireDate;
}
```

Employee

Harper Russo

Member

Leader

Активация Wind
Чтобы активировать W
"Параметры".

```java
public class User implements Serializable, Comparable<User> {
    public boolean changePassword(String newPassword) {
        if (newPassword != null && !newPassword.isEmpty()) {
        }
        return false;
    }


    @Override
    public String toString() {
        return "User{" +
                "id=" + id +
                ", firstName='" + firstName + '\'' +
                ", lastName='" + lastName + '\'' +
                ", email='" + email + '\'' +
                '}';
    }


    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (!(obj instanceof User)) return false;
        User other = (User) obj;
        return Objects.equals(getEmail(), other.getEmail()) &&
                Objects.equals(getFirstName(), other.getFirstName()) &&
                Objects.equals(getPassword(), other.getPassword()) &&
                Objects.equals(getLastName(), other.getLastName());
```

User

```java
public class FinanceManager extends Employee {
    public FinanceManager(String firstName, String lastName) {
    }

    public FinanceManager(Integer id, String firstName, String lastName, String email, String password, double sala
        super(id, firstName, lastName, email, password, salary);
    }


    public void sendSalary(Employee e) throws Exception {
        if (e == null) {
            throw new IllegalArgumentException("Employee cannot be null.");
        }
        if (this.getSalary() <= 0) {
            throw new IllegalStateException("Finance Manager's salary is not set or invalid.");
        }
        String salaryMessage = "Your salary: " + this.getSalary();
        Message m = new Message(e, salaryMessage);
        m.setSender(this);
        SaveData.INSTANCE.addMessage(m);
        System.out.println("Salary message sent to Employee ID: " + e.getEmail());
    }
```

Manager

```java
iblic class Teacher extends Employee{
    }

    public void putMark(Lesson l, Student s, Double score, int currentWeek) throws IOException {
        if (currentWeek <= 8) {
            s.getMarks().get(l.getCourse()).setAtt1(score);
        } else if (currentWeek >= 14 && currentWeek <= 15) {
            s.getMarks().get(l.getCourse()).setAtt2(score);
        } else if (currentWeek >= 16) {
            s.getMarks().get(l.getCourse()).setFinalExamScore(score);
        }
        SaveData.write();
    }

    public void addCourse(Course course) {
        if (!courses.contains(course)) {
            courses.add(course);
            course.assignTeacher(this);
        }
    }

    public List<Student> viewStudentInfo(Lesson l){
        return SaveData.INSTANCE.getLessonsOfStudents().entrySet()
                .stream()
                .filter(n->n.getValue().contains(l))
                .map(Map.Entry::getKey)
                .collect(Collectors.toList());
```

Teacher

```java
public static int calculateHIndex(Researcher researcher) {
    Collections.sort(papers, (p1, p2) -> Integer.compare(p2.getCitations(), p1.getCitations()));

    int hIndex = 0;

    // Iterate through the papers and calculate the H-index
    for (int i = 0; i < papers.size(); i++) {
        // Check if the current paper has at least (i + 1) citations
        if (papers.get(i).getCitations() >= (i + 1)) {
            hIndex = i + 1; // Update the h-index to (i + 1)
        } else {
            break; // If we encounter a paper with fewer citations than the current index, stop the loop
        }
    }

    return hIndex;
}

/**
 * Takes a researcher as input and returns a list of research papers that the researcher has authored
 * @param researcher Instance of the Researcher class that represents a researcher who has authored one or more research pape
 * @return A list of ResearchPaper objects
 */
public static List<ResearchPaper> printPapers(Researcher researcher){
    List<ResearchPaper> papers = SaveData.INSTANCE.getResearchPapers().stream().filter(n->n.getAuthors().contains(researcher)
    return papers;
```

Researcher

```java
/**
 */
public HashMap<Course, Mark> getMarks() {
    return marks;
}

public String toString() {
    return "Student [" + super.toString() + ", school= " + this.school + ", year of study= " + this.yearOfStudy + "]";
}

public boolean equals(Object o) {
    if(!super.equals(o)) return false;
    Student s = (Student) o;
    return this.gpa == s.gpa && this.yearOfStudy == s.yearOfStudy && this.school.equals(s.school);
}

public int hashCode() {
    return Objects.hash(getEmail(), getId(), getFirstName(), getPassword(), getLastName(), getSchool(), getMajor(), getGpa(),
}

/**
 * Inner class representing a student who is also a researcher
 */
public class ResearcherStudent implements Researcher, Serializable {

    @Override
    public String toString() {
```

Student

University System uniqueness:

For Students:

- Viewing grades for courses.
- Registering for courses.
- Viewing class schedules.

For Teachers:

- Viewing the courses they teach.
- Managing student grades.
- Uploading educational materials.
- Sending notifications to students.

For Administrators:

- Managing user accounts (adding, deleting, and editing).
- Viewing and editing courses.
- Assigning teachers to courses.
- Managing system settings and data storage.

For Technical Support:

- Monitoring system performance.
- Resolving technical issues for users.
- Managing data access and fixing system errors.

General System Features:

- Data persistence using serialization: Ensures that data is not lost during system failures.
- Role-based access control: Users only see the functions relevant to their roles.
- Centralized data management: All data about students, teachers, courses, and schedules is stored in one place.
- Intuitive console interface: Easy access for users with minimal technical expertise.
- Data search and filtering: For example, searching for students by name or courses by title.

This system aims to streamline key university processes, making them more efficient and less prone to errors.

.

Our system couldn't do:

Registration on FX:

Student Request

ICP(advisor)

Acadmic mobility

Conclusion

To cope with this project, we had to put in a lot of effort. But working in a friendly team, we coped with all the difficulties and went through one of the most difficult, but at the same time the most interesting stages of the project.

Thank you